Remarks

Entry of this amendment, reconsideration of the application and allowance of all claims are respectfully requested. Claims 1-58 are now pending.

By this paper, independent claims 1, 15, 20, 34, 35, 40, 41 & 51 are amended and new claims 55-58 are added to more particularly point out and distinctly claim the subject matter of the present invention. These amendments to the claims constitute a *bona fide* attempt by the Applicants to advance prosecution of this application and obtain allowance of certain claims and are in no way meant to acquiesce to the substance of the outstanding rejection. It is believed that the amendments to the claims place all claims in condition for allowance. Support for the amended and new claims can be found throughout the application as filed. For example, reference paragraphs [0011], [0027] – [0033], [0046] & [0058] – [0060] of the specification, as well as FIGS. 2-4 thereof. No new matter is added to the application by any amendment presented.

In the Office Action, claims 15-19, 34 & 40 were rejected under 35 U.S.C. §101 because the claimed invention was allegedly directed to non-statutory subject matter. Additionally, claims 1-54 were rejected under 35 U.S.C. §103(a) as being unpatentable over Passova (U.S. Patent No. 6,671,874 B1; hereinafter Passova) in view of Okayasu (U.S. Patent No. 5,659,554; hereinafter Okayasu). Each of these rejections is respectfully, but most strenuously, traversed to any extent deemed applicable to the claims presented herewith.

Regarding the 35 U.S.C. §101 rejection, claims 15, 34 & 40 are amended herein to recite various computer-implemented methods and systems. Based on these amendments, reconsideration and withdrawal of the §101 rejection is respectfully requested.

In one aspect, Applicants' invention is directed to a technique for testing a software component comprising multiple layers of software (e.g., claims 1, 20 & 41). This technique includes:

1. Creating an abstraction matrix in mathematical abstract form by partitioning the software component into multiple layers. The abstraction matrix comprises state and event information taking into account relationships that exist between the multiple software layers.

2. Parsing the abstraction matrix to automatically generate and factor out doable test cases and mapped expected results therefor.

3. Separating the test cases based on the layers of the software component and associating data structures with the separated test cases of the software layers, the data structures allowing the test cases of the various software layers to be uncorrelated.

4. Employing the software component in executable form to generate for each software layer of the software component test case execution threads from the test cases and mapped expected results for that software layer.

5. Executing in parallel at least some of the test case execution threads for at least one software layer of the software component, thereby testing the software component.

In another aspect, Applicants' invention is directed to a computer-implemented technique for generating test cases for use in testing a software component comprising multiple layers of software (e.g., claims 15, 35, 40 & 51). This technique includes:

1. Ascertaining a functional specification of a software component.

2. Creating an abstraction matrix in mathematical abstract form that describes the software component by partitioning the software component into multiple software layers using the functional specification. The abstraction matrix comprises state and event information which takes into account relationships that exist between the multiple software layers.

3. Parsing the abstraction matrix to automatically generate and factor out doable test cases and mapped expected results therefor.

4. Separating the test cases based on layers of the software component, and associating data structures with the separated test cases of the software layers. The data structures allow the test cases of the various software layers to be uncorrelated.

Advantageously, Applicants' creating of the abstraction matrix to describe the software component in mathematical abstract form by automatically partitioning the software component into multiple software layers (e.g., using the functional specification) and accounting for relationships that exist between the software layers, allows independent tests per software layer to be automatically constructed and factored out, thus significantly reducing the number of tests needed to verify a software component.

Applicants respectfully traverse the outstanding rejection on the following grounds: (1) the Office Action has misinterpreted the teachings of Passova, thus voiding the basis for the rejection; (2) the combination fails to teach or suggest one or more aspects of Applicants' invention as recited in the claims presented herewith; (3) the documents themselves lack any teaching, suggestion or incentive for their further modification as necessary to achieve Applicants' recited invention; and (4) the combination, to the extent characterized in the Office Action, is a hindsight attempt to reconstruct the claimed invention using Applicants' own disclosed subject matter.

Passova describes a method of developing a model of a computer program by representing the system requirement conditions and events as a set of positions and transitions. A universal software test development method is applied to the model by mapping the conditions to a set of positions including a starting position which has no input, a terminal position which has no output, and intermediate positions between the starting position and the terminal position. Both the model development method and the test development method can be implemented automatically by computer software, and the test development method allows for automated development of test documentation. These methods can be applied to the verification and validation of computer software. (See Abstract.)

Relative to Applicants' independent claims, Applicants initially respectfully submit that Passova fails to teach or suggest a software component partitioned into multiple layers of software *per se*, let alone via an abstraction matrix in mathematical abstract form. The Office Action does not directly address this characterization of Applicants' invention, but asserts at page 5 that Passova's Use Cases hint partitioning of a whole product going by level of requirements. This characterization is respectfully traversed. The level of requirements in Passova referred to are taught at column 7, lines 9-15 thereof to comprise three levels of the software development cycle. The software development cycle in Passova includes system requirements, design requirements and implementation requirements. These requirements and levels of software design do not equate to, nor suggest, Applicants' recited software component comprising multiple layers, let alone the partitioning of the software component into multiple software layers. Clearly, a level of software development is a different concept than the characterization recited in Applicants' independent claims. Although employed slightly differently, Applicants' specification indicates that the techniques presented herein can be used with any level of software development, i.e., unit tests, function tests and/or integration tests. For the above reasons, Applicants respectfully submit that Passova does not hint at Applicants' recited partitioning of a software component into multiple software layers. The layers of software refer to the software component itself, and not to any system level requirement, design level requirement, or implementation level requirement as discussed in Passova.

Still further, various ones of Applicants' independent claims recite automatically partitioning the software component into multiple software layers. Clearly, since there is no teaching of partitioning the software components into multiple software layers in Passova, there is no teaching or suggestion therein of a facility for automatically partitioning the software component as recited by Applicants.

Additionally, Applicants recite parsing the extraction matrix to automatically generate and factor out doable test cases and mapped expected results therefor. A careful reading of Passova fails to uncover any parsing of an abstraction matrix *per se*. In this regard, the Office Action cites column 3, lines 61-63, column 7, lines 9-15 & column 14, line 8 to column 15, line 34 of Passova. However, no explanation is provided as to how Applicants' characterized facility is being taught in these lines. The cited lines of Passova merely refer to automatically generating

tests of a program. The material does not teach or suggest Applicants' recited parsing of the abstraction matrix as part of automatically generating test cases.

Further, as recited in the claims presented, Applicants automatically generate and factor out doable test cases and mapped expected results therefor. Thus, not only are test cases being generated in Applicants' invention, but the doable test cases are automatically factored out (i.e., identified). This facility for automatic generating and factoring out doable test cases is believed clearly distinct from any teaching of Passova (or Okayasu). Conventionally, factoring out or identifying "doable" test cases has been a manually intensive operation.

Still further, Applicants recite separating the test cases based on the software layers of the software component, and associating data structures with the associated test cases of the software layers. The data structures allow the test cases of the various software layers to be uncorrelated. In this regard, the Office Action references the requirement level/use cases of the software component under design in Passova, and in particular, column 7, line 6 – column 8, line 15 & Tables 5-6 of column 14 and Table 7 of column 15. Applicants respectfully traverse this characterization of the teachings of Passova. As indicated above, the levels and the requirements referred to in Passova refer to the software development cycle levels, and include system requirements, design requirements and implementation requirements. Therefore, the concept of separating test cases based on different levels in Passova does not equate to or suggest Applicants' recited separating of test cases based on software layers of the software component. Further, Applicants recite that the data structures associated with the separated test cases allow the test cases of the various software layers to be uncorrelated. No similar functionality is recited by Passova since the use case flow of Passova refers to the development cycle, and not to different layers of the software component to be tested.

For at least the above reasons, Applicants respectfully submit that the independent claims presented herewith patentably distinguish over the teachings of Passova, alone or in combination with Okayasu.

Okayasu teaches a test case generating apparatus for generating test cases used in verifying operations of an object. The apparatus includes: an operation specification describing unit for describing the operation specifications of the object using a plurality of parallel and/or hierarchical state transition diagrams; a state transition diagram searching unit for searching the plurality of state transition diagrams described by the apparatus specification describing unit to select a transition from a predetermined state to a state on the state transition diagram, and outputting a selection result; and an action/event relation managing unit for storing relations between actions and events in controlling a transition selection operation of the state transition diagram searching unit by managing transitions which never occur in practice. The apparatus further includes a parallel (hierarchy) relation managing unit for controlling the transition selection operation of the state transition diagram searching unit by managing the searching order of the plurality of parallel and/or hierarchical state transition diagrams described by the operation specification describing unit. (See Abstract.)

Okayasu is cited in the Office Action for teaching organizing state transition diagrams in a hierarchy diagram representing components that connect in parallel with each other. Without acquiescing to this characterization, Applicants respectfully submit that a careful reading of Okayasu fails to uncover any teaching or suggestion of a technique for creating an abstraction matrix which includes automatically partitioning a software component into multiple software layers. In Applicants' recited invention, a facility is providing for automatically partitioning the software component. No similar capability is presented by Okayasu (or Passova). For at least this reason, Applicants respectfully submit that their independent claims patentably distinguish over the applied art.

Further, Applicants respectfully submit that a careful reading of Okayasu fails to uncover any teaching or suggestion relevant to the above-noted deficiencies of Passova when applied against the independent claims presented. For example, Okayasu fails to teach or suggest parsing an abstraction matrix in mathematical abstract form used to automatically generate and factor out doable test cases and mapped expected results therefor, or separating the test cases based on the software layers of the software component, and associating data structures with the separated test cases of the software layers, the data structures allowing the test cases of the various software layers to be uncorrelated.

For at least these reasons, Applicants respectfully submit that the recited independent claims would not have been obvious to one of ordinary skill in the art based upon the teachings of Passova and Okayasu.

Still further, upon a review of the applied patents, there is no teaching, suggestion or incentive for further modification of the combination as would be necessary to achieve Applicants' invention. In this regard, neither patent teaches creating an abstraction matrix in mathematical abstract form by automatically partitioning a software component into multiple software layers. Further, neither patent teaches parsing the abstraction matrix to automatically generate and factor out doable test cases and mapped expected results therefor. Still further, neither patent teaches separating the test cases based on the software layers of the software component, and associating data structures for the separated test cases of the software layers, the data structures allowing the test cases of the various software layers to be uncorrelated.

Yet further, various characterizations of the teachings of Passova and Okayasu provided in the Office Action set forth no technical basis outside that contained in Applicants' own specification for the functionality at issue. Various characterizations of the teachings of Passova in particular merely assert the language of Applicants' claimed invention in hindsight. For example, reference page 4 of the Office Action concerning Applicants' recited facility for parsing the abstraction matrix to automatically generate and factor out doable test cases and mapped expected results therefor. Thus, the rejection is believed to violate the well known principle that Applicants' own disclosure cannot used as a reference against them.

The consistent criterion for the determination of obviousness is whether the art would have suggested to one of ordinary skill in the art that the claimed invention should be carried out and would have a reasonable likelihood of success, viewed in light of the prior art. The suggestion and the expectation of success must be found in the prior art, not in Applicants' disclosure. At various points of the Office Action, the alleged combination at issue is simply characterized in the language of Applicants' own disclosure, rather than an identified basis in the prior art for achieving the modifications necessary to arrive at Applicants' claimed invention, in violation of this well known principle. This is yet another, independent reason why the current invention is not obvious over the applied art.

In summary, Applicants traverse the rejection of the independent claims presented based upon the misinterpretation of the Passova and Okayasu patents; the lack of a teaching of one or more aspects of Applicants' invention based on the combination; the lack of an actual teaching, suggestion or incentive in the art for the modifications necessary to achieve their invention; and the combination, to various extent characterized in the Office Action, is a hindsight reconstruction of the claimed invention using Applicants' own disclosed subject matter.

For all the above reasons, Applicants respectfully submit that independent claims patentably distinguish over the teachings of Passova and Okayasu. Reconsideration and withdrawal of the obviousness rejection based thereon is therefore respectfully requested. The dependent claims are believed allowable for the same reasons as the independent claims, as well as for their own additional characterizations.

Thus, Applicants respectfully submit that all claims are in condition for allowance, and such action is respectfully requested.

If a telephone conference would be of assistance in advancing prosecution of the subject application, Applicants' undersigned attorney invites the Examiner to telephone him at the number provided.

Respectfully submitted,

Kevin P. Radigan

Kevin P. Radigan
Attorney for Applicants
Registration No.: 31,789

Dated: April 12, 2005.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579